Picsellia

WHITE PAPER

An Elementary Introduction to MLOps For Computer Vision

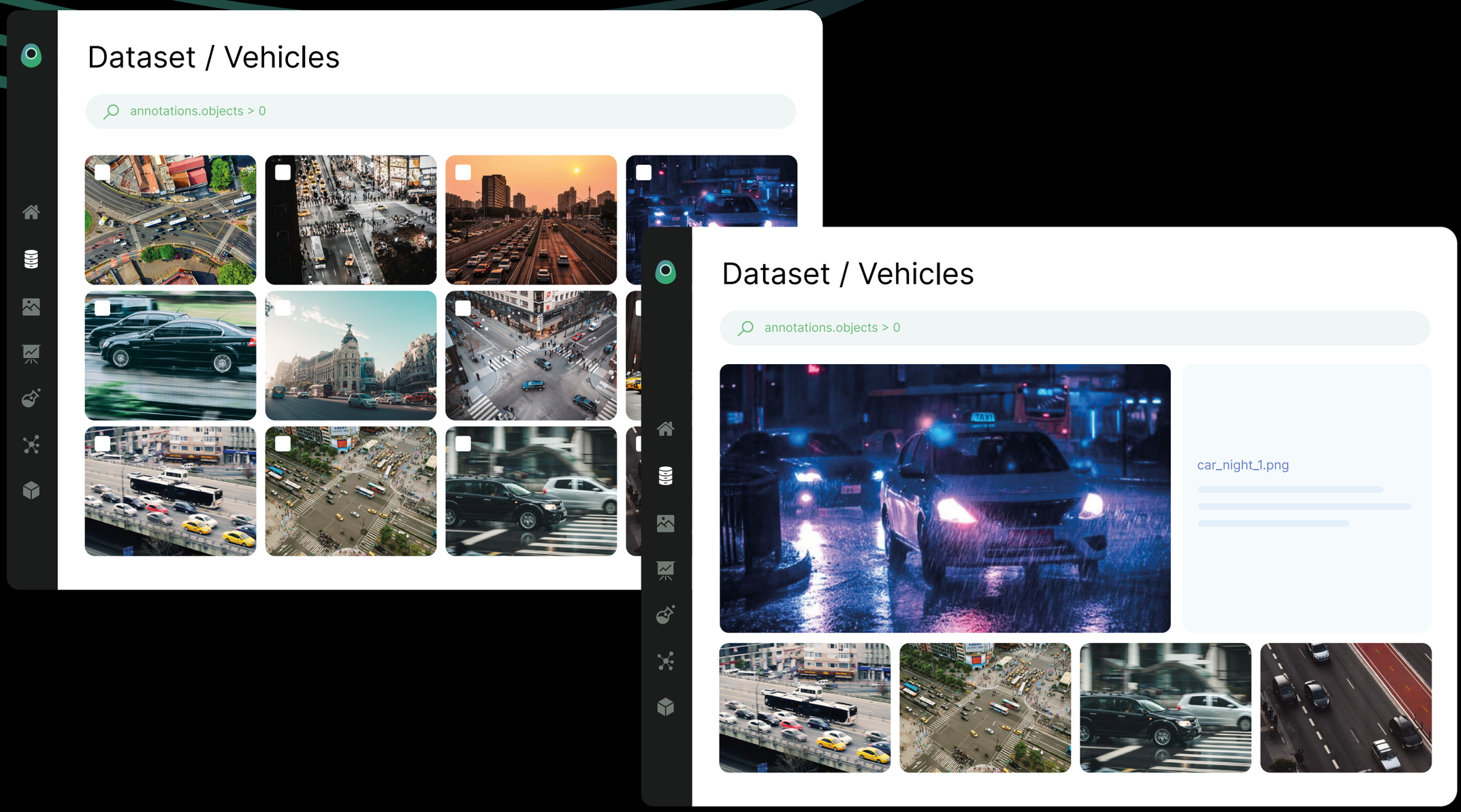


Table of contents

CHAPTER 1

Introduction to MLOps

The Rise of MLOps

MLOps or Machine Learning Operations–a concept that has been around since 2015, started becoming popular in 2020 when businesses began facing challenges in operationalizing machine learning (ML) models at a large scale.

In 2019, VentureBeat reported that 87% of ML projects were not making it into production due to a lack of leadership, high-quality data, and organizational siloes. This leads to many issues related to ML security, monitoring, scalability, auditing, bias, and eventually, failure of the ML project.

A survey, "State of MLOps in 2021," indicates that respondents showed increasing interest in developing, deploying, and monitoring machine learning models for production environments. Today, the global MLOps market size stands at a little less than a billion dollars, but it is expected to rise considerably and grow to more than USD 6 billion by 2028 at a CAGR of 39.07% between the forecast period of 2022 to 2028.

AI practitioners are focused on more than just data analysis and state-of-the-art algorithms. They are taking a significant interest in enterprise-grade machine learning operationalization and productionization.

Most AI & ML market leaders are now offering MLOps as part of their product offerings to **strengthen the capabilities of their machine learning platforms.** Overall, the AI ecosystem is witnessing the rise of MLOps strategies among AI practitioners to ensure the success of their AI initiatives.

Machine Learning Development Lifecycle and the Role of MLOps

The machine learning development lifecycle is analogous to the software development lifecycle (SDLC). However, ML relies on data and algorithms. So, both aspects must be managed effectively throughout the ML pipeline.

A typical ML development life cycle involves the following stages:

1. Problem understanding and requirement gathering
2. Data gathering and annotation
3. Data exploration & analysis (also known as Exploratory Data Analysis (EDA))

4. Feature engineering
5. Model training
6. Model testing
7. Model deployment

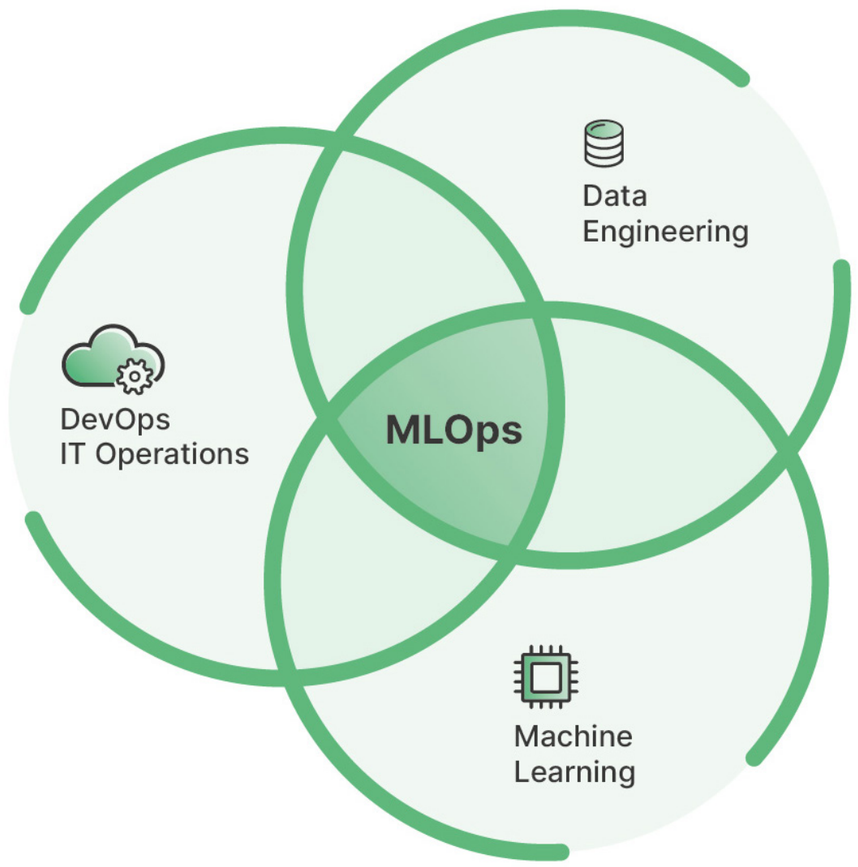
These steps would suffice for a small-scale ML application. But what if the application users grow dramatically?

Mass adoption of AI among end-users and higher competition among AI vendors has completely changed the AI landscape. Today, the network traffic or requests coming into the AI application is exponentially times greater compared to a few years back.

In such circumstances, the deployed ML models must be retrained on new data to expand, adapt, and scale. This leads to more experimentation and several versions of ML code and data, which needs effective version control. As a result, the overall process becomes time-consuming and tedious.

This is where MLOps steps in. Inspired by DevOps practices, it is an intersection of machine learning, data engineering, and IT operations.

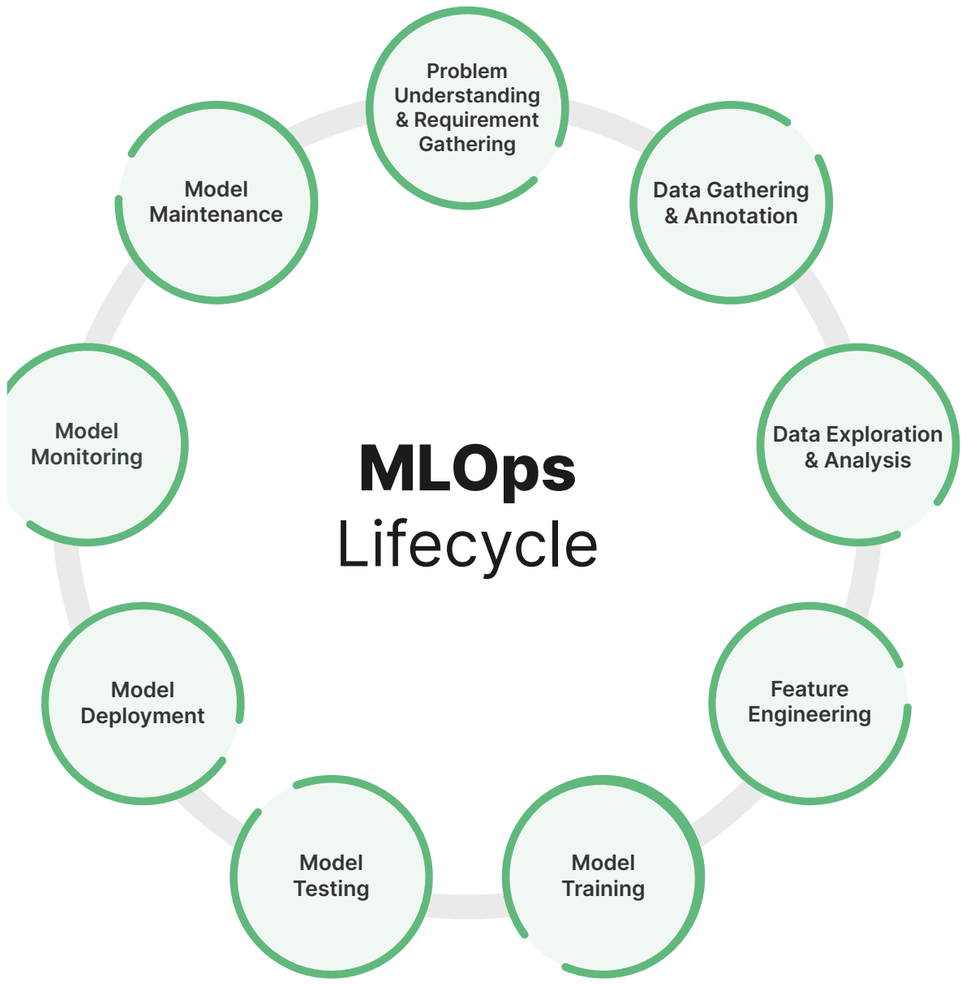
Just as DevOps has increased SDLC's development and deployment velocity, MLOps simplifies large-scale ML deployment complexity by automating various steps in the ML lifecycle.



MLOps domain intersection

To facilitate an enterprise-grade ML initiative, MLOps adds two more stages in the abovementioned development lifecycle:

- Model monitoring
- Model maintenance



The complete MLOps lifecycle

Generally, MLOps allows AI teams to monitor production pipelines automatically and notify stakeholders if any issues are detected. However, there are many challenges in implementing MLOps strategies, which we'll discuss in the next section.

Challenges in MLOps Implementation

The process of MLOps needs to begin with the end-user in mind. Initially, the main challenge is to interpret the business problem statement accurately. As a decision-maker or solution architect, ask these questions:

- What are the end-user's pain points?
- What is it that the users value the most in an ML application?
- How will end-users interact with the ML system?
- How do the end-users requirements map to the ML domain?
- Is the relevant data available?
- What metrics determine a successful ML model?
- What are the limitations of the existing ML technology, and how will it affect the end product?

Asking relevant questions rules out unrealistic expectations. This means the company needs to work on its customer-centricity–relationship with its users–so that they are willing to share relevant information whenever the business needs more insights. Moreover, AI practitioners must carefully set the project's success metrics and KPIs

based on business requirements. If the project tracks wrong success metrics due to poor requirements analysis, then the project design, including feature engineering, model training, and testing, would be flawed.

During development, an MLOps pipeline goes through the following processes:

- **Continuous Integration (CI):** Automatically integrates application code changes from multiple contributors into a central repository.

- **Continuous Delivery (CD):** Automatically deploys (or releases) software in production from the central repository.

- **Continuous Training (CT):** Automatically retrains and deploys ML models in production with new data.

Each of these processes presents several MLOps challenges. An inefficient CI/CD implementation leads to slow server responses, excessive memory usage, and performance issues. Moreover, machine learning models get stale with time due to data or distribution drift (discussed in Chapter 2). This is where continuous training is significant. However, an inefficient retraining strategy fails to reduce the production model's degradation. Also, the computational and labor costs of CT can increase unexpectedly if it is not systematically monitored. In most cases of inefficient implementation, issues are not detected and communicated to the team promptly, which hinders quick resolution, resulting in loss of revenue and customer dissatisfaction.

Another challenge in implementing the MLOps pipeline is misaligned or out-of-sync version control–for both model and data, which can break the pipeline due to compatibility issues. To ensure smooth implementation, the application testing mechanism must check if everything is working. If the testing system is somehow flawed, it could lead to the deployment of faulty models, which is a nightmare for the MLOps pipeline.

Once the model is in production, the MLOps pipeline becomes more vulnerable. Generally, the deployed models process real-world enterprise data of different types – structured, unstructured, or semi-structured, coming from various sources. If the incoming data differs from the data on which the model is trained, the model could yield unexpected prediction outcomes, eventually leading to model decay and deeming it unfit to serve consumers.

Moreover, in production, the incoming network load for the deployed model can increase drastically. The entire application can face downtime if the pipeline is not programmed to scale automatically. Even when you try to scale operations, if models are not built not withstand extendibility, then the pipeline will fail.

MLOps or Machine Learning Operations–a concept that has been around sinceMaintaining high security is another concern in the MLOps pipeline. ML models are prone to security threats, such as

- Training data poisoning, i.e., tampering with the training dataset by adding malicious data points or samples to alter the prediction behavior.

- A well-designed backdoor in the training data produces malicious output. A backdoor attack would leave most of the training data untouched. It will only corrupt a small portion of the data to control the model's output as desired by the attacker.

- Adversarial attacks using test inputs to bypass or mislead models with malicious instances.
- Model theft, particularly model parameters, to extract information about the training data or reconstruct the target model.
- Theft of sensitive training data using model inversion or membership inference attacks. In model inversion attacks, the attacker successfully extracts diverse samples for each class of the training dataset. While membership inference attacks allow the attacker to query the training dataset to determine whether a particular sample is contained within the training data.

Most of these security vulnerabilities can go unnoticed due to the unexplainable nature of the ML models. MLOps offers a systematic approach to identifying and resolving such issues.

In the entire MLOps pipeline, any number of factors could go wrong. It is important to be aware of all the implementation challenges and address them proactively to ensure the AI application's success for its entire lifecycle.

Before we discuss MLOps further, let's see how it compares with DevOps.

MLOps vs. DevOps

MLOps and DevOps simplify, accelerate, and automate software engineering processes, where DevOps focuses on application development, and MLOps–a specialized subset of DevOps, deals with machine learning development and predictive intelligence.

Both processes aim to speed up the deployment and delivery of their respective products. But the machine learning lifecycle introduces some new nuances that are generally not addressed in the DevOps approach.

Here are the six main aspects where MLOps differs from DevOps:

1. **Specialized Resources**
   The DevOps culture is spearheaded by software engineers specializing in the relevant development stack, along with QA engineers, cloud architects, and networking resources. Out of these, a senior engineering resource often assumes the role of the DevOps engineer, who is usually aware of all aspects of the application code and deployment pipeline. However, companies also hire specialized DevOps resources with proven industry experience to streamline their software application delivery.
   On the other hand, an MLOps team consists of specialized data scientists, data analysts, ML engineers, ML architects, and MLOps engineers. (More on this in the next section).

2. **Continuous Training Pipeline**
   MLOps is built on the same DevOps principles. But it has an added continuous training (CT) process

specific to the ML domain. ML projects have two main components that keep changing – the code and the data. Hence, any ML model deployed in production must be continuously updated using new real-world data. The CT pipeline automatically ensures that the production model maintains and improves performance.

3. **CI/CD Pipeline**
   In DevOps, the CI/CD pipeline mainly deals with the development and deployment of core builds and modules. On the other hand, MLOps deals with models. Model training, testing, and validation are specialized MLOps processes. In both practices, pipelines are configured to test the builds of their respective applications. DevOps executes unit and functional tests, while MLOps evaluates the pipeline on various model testing scenarios.

4. **Continuous Monitoring**
   MLOps puts data first. It uses data insights to improve ML models' development and deployment processes. It tracks various metrics that observe a model's performance in production. Some of these metrics include precision, recall, f1-score, MAE, RMSE, etc. The data-first approach enables lesser human intervention.
   On the other hand, DevOps puts people first. It improves communication between IT and operations departments and breaks down organizational silos by automating software processes. Generally, it monitors the software application for regular maintenance and leaves the rest to the stakeholders.

5. **Version Control**
   DevOps pipeline only manages the changes in the application code and artifacts. On the other hand, MLOps tracks all training/testing experiments, which includes all versions of the model, weights, and hyperparameters. This process is known as ML experiment tracking (discussed in Chapter 2).
   Moreover, MLOps pipelines manage data versioning. It's important because each data version corresponds to different ML models. Data versioning enables ML practitioners to reproduce and trace experiments.

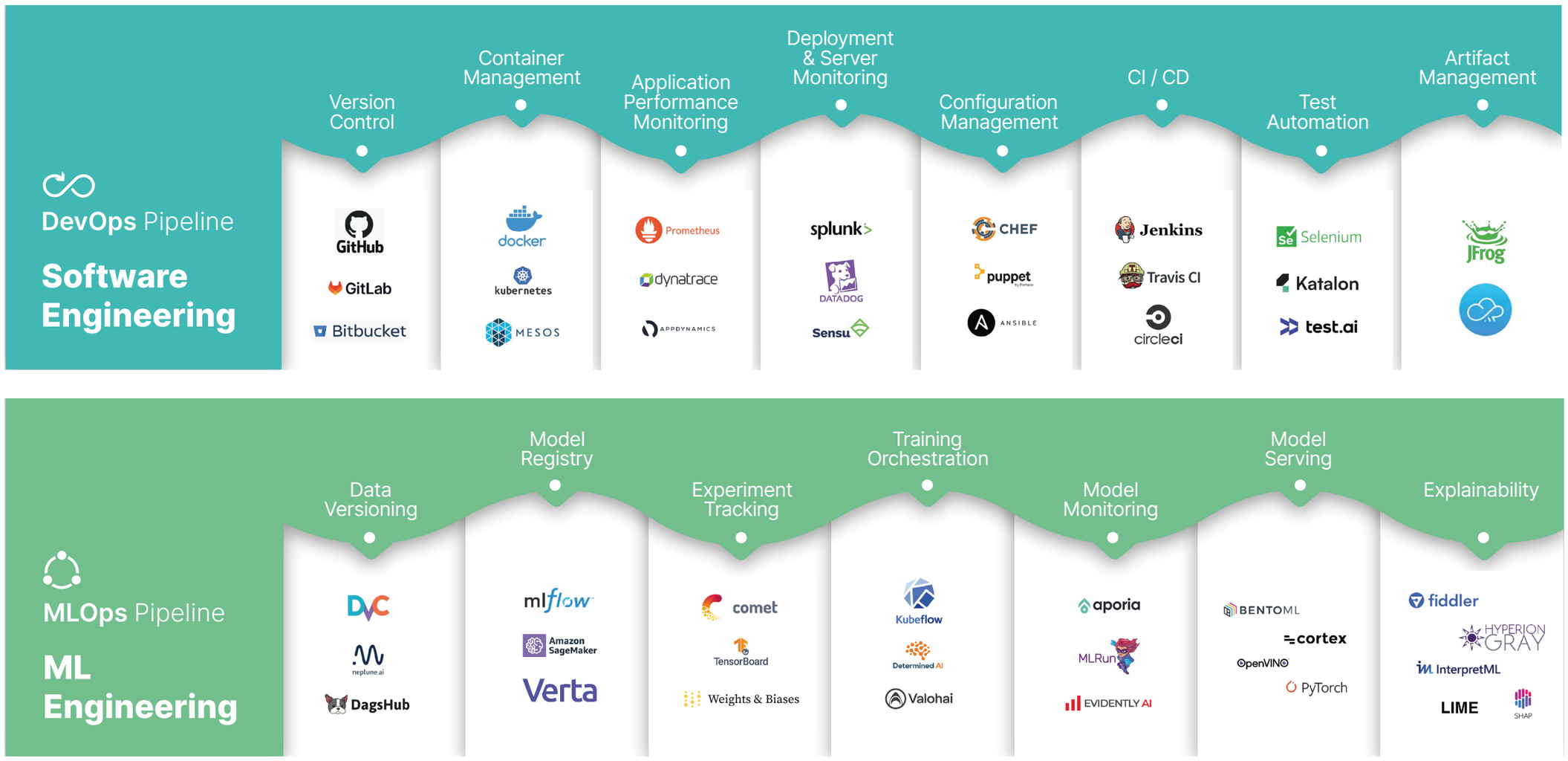
6. **Infrastructure & Automation Tools**
   A DevOps pipeline contains various tools related to different stages of software development. These include:

- Version control tools like GitHub, GitLab, and Bitbucket
- Container management tools like Docker, Kubernetes, and Mesos
- Application performance monitoring tools like Prometheus, Dynatrace, and AppDynamics
- Deployment and server monitoring tools like Splunk, Datadog, and Sensu
- Configuration management tools like Chef, Puppet, and Ansible
- CI/CD tools like Jenkins, Travis CI, and CircleCI
- Test automation tools like Selenium, Katalon Studio, and Test.ai
- Artifact management tools such as JFRog Artifactory and CloudRepo

On the other hand, an MLOps pipeline constitutes various tools related to machine learning development and deployment. These include:

- Data versioning tools like DVC, Nepture, and DAGsHub
- Model registry tools like MLflow, Amazon SageMaker, and Verta.ai

- Experiment tracking tools like Comet, TensorBoard, and Weights & Biases
- Training orchestration tools like Kubeflow, Determined, and Valohai
- Model monitoring tools like Aporia, MLRun, and Evidently AI
- Model serving tools like BentoML, Cortex, OpenVINO Model Server, and TorchServe
- Explainability tools like ELI5, Fiddler, InterpretML, Lime, and SHAP



A illustration of DevOps and MLOps tools ecosystem

MLOps and DevOps pipelines offer a cooperative effort and enable cross-team collaboration. However, these pipelines must be adapted according to business and domain requirements. Generally, each domain contains unique technical nuances that must be addressed using specialized processes. Computer vision (CV) is one such domain that requires a specialized MLOps approach. Let's discuss it further.

Applying MLOps to Computer Vision–Challenges & Limitations

With the advent of Iot and sophisticated imaging devices, AI and ML are now largely being used to analyze images to develop advanced computer vision applications. For instance, medical imaging data is processed by CV models to develop various AI-powered healthcare applications like cancer detection, brain tumor detection, fall detection, accurate machine-assisted medical diagnostics, surgical simulations, remote health monitoring, etc.

When it comes to applying MLOps to CV tasks, certain challenges must be addressed to streamline their development and deployment. Some of these challenges are:

- Computer vision deals with highly unstructured image and video data, while regular AI applications generally deal with text-based data.
- The size of the CV data is significantly larger than text-based data. Hence, regular data tools cannot manage the sheer computer vision data volume.
- Image or video data require annotations, which is a daunting task considering the size of the data.
- CV models require high-quality domain-specific data to achieve good performance. For instance, a CV model would unlikely differentiate between a real car and a toy car if the car dataset is not cleaned up and annotated properly.
- Image or video streams are often captured by multiple imaging devices, which must be aligned with the CV application to achieve favorable results.

Hence, due to these and many other challenges, traditional MLOps cannot be directly applied to CV. Computer vision requires specialized and tailor-made MLOps pipelines. To address these challenges, this white paper discusses (in Chapter 2) various aspects of CVOps–a specialized set MLOps practices applied to computer vision projects.

The People of MLOps

Since MLOps is a merger of DevOps, machine learning, and business operations, a successful pipeline requires collaborative efforts from various resources.

Let's look at each role that is crucial to the success of the ML life cycle.

- **Subject Matter Experts (SMEs)**
  SMEs understand the business domain from a machine learning perspective. They work with data scientists to translate business requirements into AI features. Through metrics, goals, and KPIs, they evaluate the entire ML life cycle and provide feedback on the performance of every process.
- **Data Scientists**
  They are mainly responsible for building, testing, and deploying ML models. They operationalize these models and maintain them in production. In case of any development and deployment issues, they are responsible for resolving any queries. In an MLOps pipeline, data scientists have the most crucial role.

- **Data Engineers**
  In large-scale applications, data generally comes from disparate sources, which is often challenging to manage. Data engineers prepare and manage these data pipelines to facilitate model building.

- **Software Engineers**
  Traditional software engineers are a crucial part of the ML pipeline because ML models or experiments are often not available in the form of standalone applications. They are usually a part of the organization's larger product offering. Hence, software engineers don't directly build ML models, nor do they have the required technical skills. But MLOps bridges the gap between software engineers and data scientists, allowing them to collaborate on the software side of the ML application.

- **DevOps Engineers**
  Generally, MLOps is integrated as part of the larger DevOps strategy in an enterprise software application. DevOps resources automate ML processes. They mainly ensure the performance, availability, and security of the ML models by working in tight collaboration with data scientists and engineers.

- **Model Risk Manager or Auditors**
  For highly-regulated industries, such as healthcare, ML applications must ensure privacy and regulatory compliance. This is where model risk managers play a crucial role. They test the application's compliance robustness during each stage of the ML lifecycle.

- **Machine Learning Architects**
  While some organizations also call them data architects, ML architects are responsible for building flexible and scalable ML production environments. They coordinate with all stakeholders, allocate resources, identify bottlenecks, and proactively find system flaws to ensure optimal performance of the ML application.

- **Chief Data Officer or Chief Information Officer**
  A CDO/CIO is responsible for building the organization's overall MLOps strategy and culture. At the executive level, this role ensures governance and sustainability of the MLOps pipeline.

MLOps pipelines are managed by a diverse group of experts who fall under these roles. At the operational level, some of these roles overlap, where an experienced resource is given multiple responsibilities, especially in small to medium-sized organizations. For instance, the role of the CDO/CIO can be performed by an experienced ML architect. However, in large enterprises, each role is explicitly defined to streamline ML processes and ensure the success of large-scale machine learning applications in the production environment.

CHAPTER 2

MLOps Toolchain for Computer Vision

Productionizing machine learning models requires effective implementation of the complete ML lifecycle while addressing various issues, including dataset dependency, complex pipelines, team collaboration, scalability, and production risks. For vision models, operationalization becomes even more complex (challenges mentioned in Chapter 1); hence CVOps plays a vital role in its successful execution. This chapter outlines the critical components that make up an effective MLOps toolchain for computer vision tasks.

1. Data Engineering & Management

Productionizing machine learning models requires effective implementation of the complete ML lifecycle while addressing various issues, including dataset dependency, complex pipelines, team collaboration, scalability, and production risks. For vision models, operationalization becomes even more complex (challenges mentioned in Chapter 1); hence CVOps plays a vital role in its successful execution. This chapter outlines the critical components that make up an effective MLOps toolchain for computer vision tasks.

Data is the most crucial element determining a project's success. For CV applications, data management becomes even more complex as the nature of data is mostly unstructured. Typically, it is available as images or videos and is considerably larger than the text-based dataset.

A CVOps tool should ensure that data is easily accessible, updatable, reusable, and securely stored. The following components are essential for building effective CV data delivery pipelines.

- **Data Repository:** Vision data should be stored in a centralized repository to encourage democratization among the internal teams. External image or video data sources should be compatible with the company's central database infrastructure. Moreover, the data repository should be encrypted and secured with robust identity and role management, along with a backup mechanism in place to support recoverability.

- **Data Organization:** Tools utilized in the CVOps pipeline should allow easy data organization with an appropriate labeling mechanism. Particularly, if media files are coming in real-time, then the CVOps pipeline should be able to automatically categorize or annotate the incoming stream and place the files in their relevant directories. The pipeline should allow tracking data changes using version control. Moreover, the company must have a pre-defined dictionary for every pipeline process to encourage standardization.

- **Navigation & Searchability:** The sheer scale of computer vision data requires a comprehensive mechanism for searching images and videos. For instance, a medical imaging dataset could have millions of images related to different parts of the human body. Each scan would correspond to a unique disease or diagnostic test. Ideally, the CV data management tool should have a tree-like navigation structure and Googlesque search capabilities, enabling teams to find and categorize the relevant resources promptly.

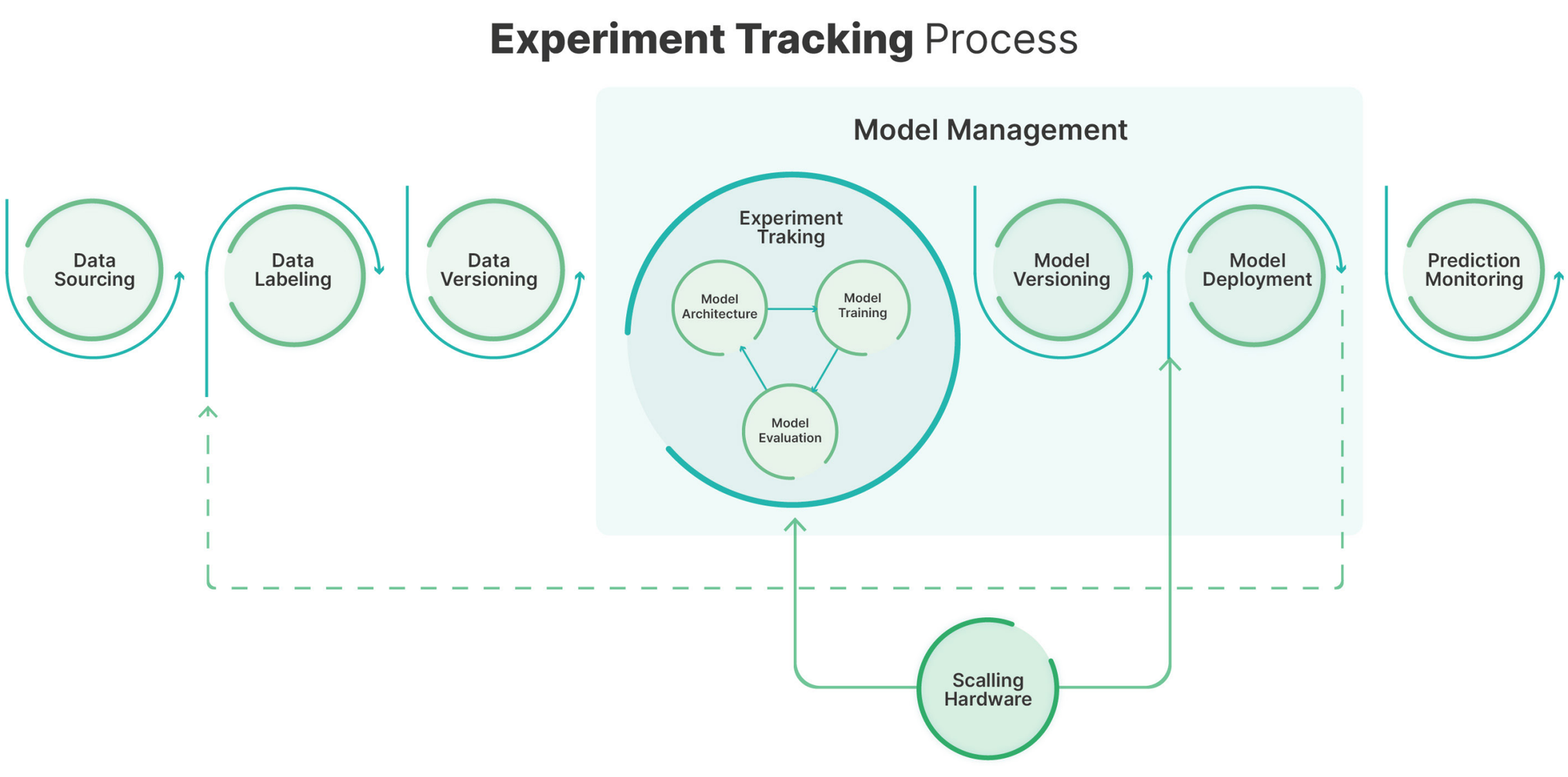
- **Metadata Management:** Data catalogs and metadata management tools are essential to any computer vision pipeline to store and manage metadata systematically. Typically, image or video metadata includes camera details like aperture, ISO number, shutter speed, camera brand, model, image/video creation time and date, and many more. Moreover, the models that have been trained on different datasets, the results of those models, the hyperparameter settings, the configuration settings, etc., must be stored by such tools to facilitate data lineage, provenance, and governance for CV pipelines.

2. Experiment Tracking & Model Building

A robust CVOps pipeline focuses on building better models that can perform consistently in production environments. To facilitate this process, data scientists perform experimentation known as machine learning experiment tracking (discussed briefly in Chapter 1).

Typically, for a given CV problem, the data scientist sets up a development pipeline to clean, explore and analyze the dataset. Then relevant data features are extracted to initiate the model-building phase, where the data scientist implements an appropriate model for training. If the model performs well on the evaluation metrics, it is pushed to the deployment pipeline.

However, it is unlikely that the data scientist will get it right the first time. Whether it is a simple image classification task or a complex segmentation task involving state-of-the-art algorithms, it requires many iterations to obtain the correct model and features. Hence, data scientists conduct a series of experiments to refine the model architecture by optimizing its parameters. In each experiment, the model is retrained and reevaluated, and the outcomes of these experiments are compared. Finally, the best-performing model is deployed in production.



A machine learning experiment tracking process illustration

To ensure seamless experimentation, a CVOps pipeline should contain a robust tracking mechanism (tool) where each run of the model-building process is traceable through a versioning system. Such a tool can generally save and compare the model performance, hyperparameters, and configurations for different ML experiments.
In addition, a CV-based experiment tracking tool ensures re-usability. The saved experiments (and their corresponding models) can be used in similar CV tasks. Team members can tweak the saved models to address similar problems, saving time and money.

3. Model Deployment

An effective CVOps pipeline can reduce deployment inefficiencies. The deployment process should ideally involve a series of tests that ensure the model is working properly before it can go out in the real world. However, the real deployment challenge lies in what type of infrastructure should be used: cloud or edge. Let's have a look below:

- **Cloud Deployment:** Typically, in a cloud environment, deployments are executed via a centralized mechanism. For instance, a CV model is deployed on a cloud server. End-users send requests to the central server via an API to get model outcomes such as predictions or recommendations. The requests (relevant computations) can be processed by the server (model) in batches or real-time, and the outcomes are returned to the end users' devices.

- **Edge Deployment:** In an edge environment, models are typically deployed on end-user devices, such as phones and tablets or local edge servers that are regionally closer to the end users. The models process local data input using the compute power of the device. A major deployment advantage is that the model can be scaled to millions of users without incurring high costs. Moreover, latency is not an issue anymore since predictions are delivered instantly.

However, with so many edge devices, the environment and infrastructure vary. This poses computability and compatibility issues and can adversely affect model performance. But, with frequent model updates, edge deployment can be highly valuable in terms of cost and efficiency.

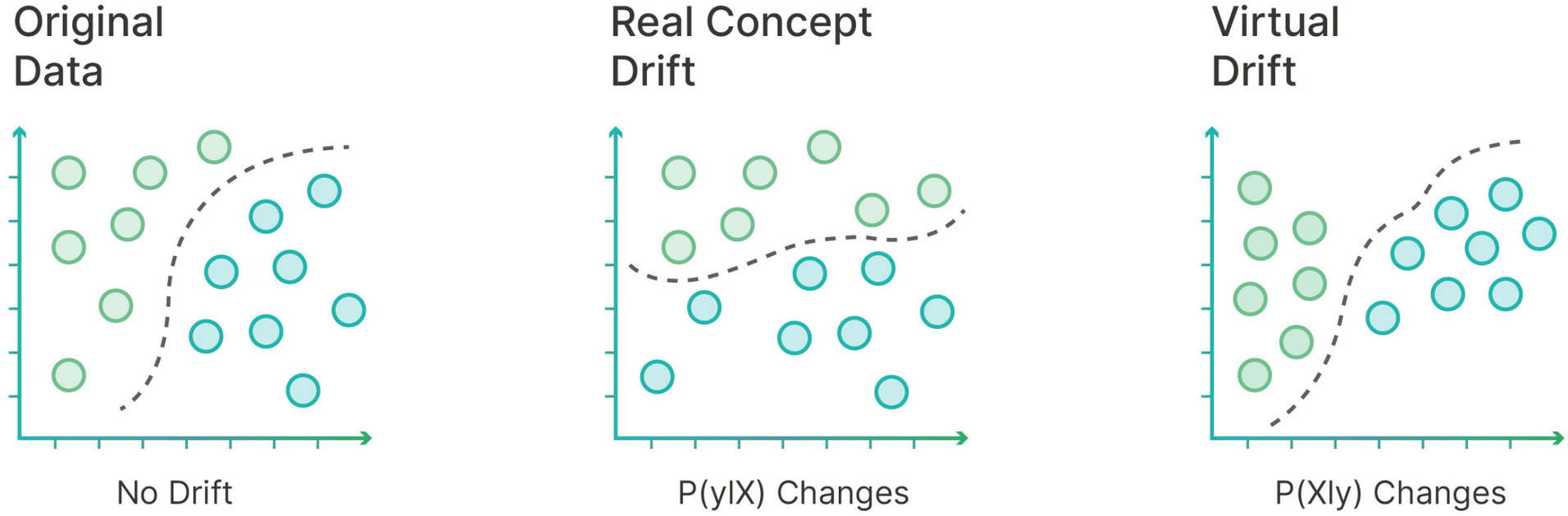
A 2021 survey by Red Hat and Pulse indicates that 78% of enterprise ML projects are deployed using a hybrid cloud strategy, including edge deployment. And only 18% are deployed solely on the cloud. Hence, edge deployment is more favorable as it conveniently facilitates AI productionization on a large scale. However, AI deployment decisions must be kept aligned with the organization's business requirements.

4. Monitoring & Feedback Loop

Model building is not a one-way road. As emphasized earlier, CVOps requires continuous training (CT) to ensure that model performance does not decay over time. Model degradation mainly occurs due to a phenomenon known as model drift, which is of two types: data drift and concept drift.

- **Data Drift:** Data drift mainly occurs when the deployed production model processes actual real-world data that is different from the data on which the model was trained, leading to performance degradation. It can occur due to training data selection bias, seasonal changes in the data, or a faulty input instrument. It does not necessarily change the decision boundary of the model.

- **Concept Drift:** Concept drift occurs when the underlying relationship between the input data and the output labels changes, mainly due to an external process or event. That is, the features developed from the training data are no longer good predictors of the target labels. Concept drift can be sudden, gradual, incremental, or recurring. Mainly, it changes the decision boundary of the model.



Illustrating the difference between data and concept drift

In CV tasks, monitoring drift can be a defining success factor. A good CVOps toolkit can proactively detect such drifts so the team can take prompt action. For instance, a CVOps pipeline can frequently compute evaluation metrics of deployed CV models and compare their values against baseline metric thresholds to detect any major deviation and send automated alerts to the team for mitigation.

5. Automated Pipeline

With manual ML development and operations mechanisms, whenever something changes, the team may not detect it quickly, resulting in model performance degradation. As a result, the users would have to bear low-quality predictions or recommendations until the model is upgraded – or worse, the model pipeline breaks completely.

A fully automated CVOps pipeline (discussed in Chapter 3 in detail) can effectively optimize every process of the ML lifecycle. All steps, from data gathering, and data validation to model building, deployment, and monitoring, can be automated to ensure a consistent model performance. The CVOps toolchain should therefore consist of an end-to-end pipeline, allowing the team to detect anomalies promptly and spend less time re-training and re-validating models manually, resulting in fast deployments.

CHAPTER 3

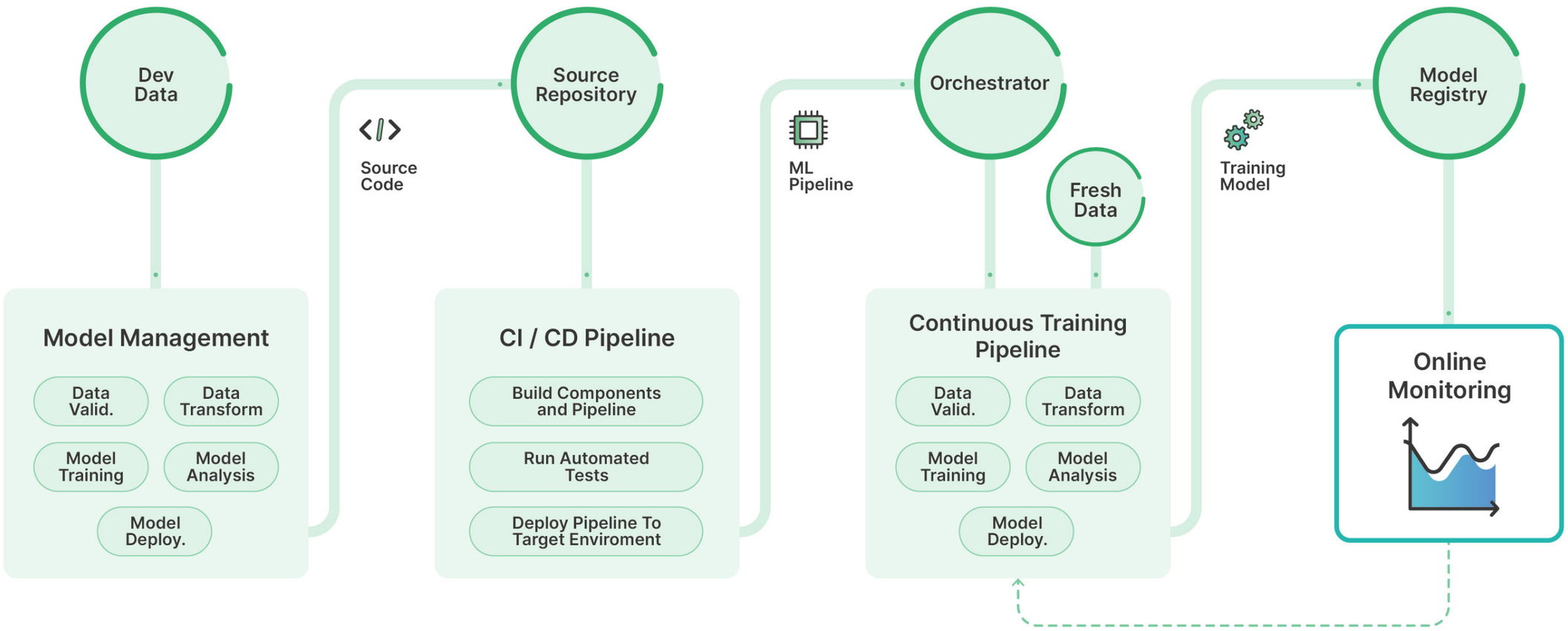
End-to-End Repeatable MLOps for Computer Vision

For successful end-to-end operations, a CVOps pipeline must contain the following characteristics:

- Flexibility
- Interoperability
- Compatibility
- User-friendliness
- Community or Vendor Support.

Let's discuss each of them in the section below.

Automated Pipeline and Compartmentalization



An illustration of an automated machine learning pipeline

Primarily, any enterprise ML pipeline should automate various steps in the ML lifecycle. Such automation is usually achieved by compartmentalizing each step as a stand-alone service rather than building a monolithic Jupyter notebook or script with all the steps implemented in a single programming module.

For instance, in an enterprise computer vision application, data ingestion can be managed by an independent service supervised by a separate team of data annotators and ETL experts. This service can then communicate with the data cleaning, wrangling, and validation service through an API maintained by a separate data team. The highly skilled team of computer vision experts can utilize the processed data pipeline to build, train, and evaluate CV models. Finally, specialized CVOps experts with CI/CD skills can build a service to deliver the model in production and monitor it around the clock. The pipeline is, therefore, a collection of such services that run independently but operate in a coordinated manner to ensure that an AI model works seamlessly. This process is analogous to working with containers in software development. Doing so with machine learning would save an organization an incredible amount of time and money while minimizing inconsistencies and avoiding duplications.

Now, the real question is: whether an organization should build such a complex pipeline from scratch, opt for vendor-managed MLOps tools, or leverage open-source frameworks. Each option needs to be evaluated against the abovementioned characteristics to avoid common MLOps pitfalls and make an informed business decision.

- **Flexibility:** Build or buy – the platform must be flexible enough to cater to different situations and edge cases. Each case may require different modeling techniques or libraries, along with varying data requirements. A custom-built platform offers the flexibility to add more features and modules to the CVOps tools in-house. But a managed CVOps platform is more rigid to any change and can only be added by the vendor.

- **Interoperability:** The market for MLOps tools is still evolving. An open-source tool that automates data ingestion and data cleaning may not support training or deployment. However, the open-source tool should have the ability to operate smoothly with the custom-built training and deployment modules. On the other hand, a managed MLOps tool is typically not interoperable.

- **Compatibility:** An effective CVOps tool must integrate with the organization's existing set of business tools. A custom-built pipeline can ensure compatibility across the stack, but it costs a lot. Also, the required talent may not be at the company's disposal. They would have to execute an expensive hiring cycle to onboard high-quality resources that can execute pipeline implementation. On the other hand, a managed CVOps tool may not offer relevant connectors for a company's existing stack by default.

- **User-friendliness:** The tool's user experience and design should be seamless to enable ease of use for different team members. In a custom-built CVOps pipeline, the team has full control over the UX elements. They can adapt it as per internal company guidelines and preferences. However, managed tools are designed by vendors to offer a generic UI and UX experience, which may not be suitable for specific business requirements.

- **Community or Vendor Support:** One of the advantages of open-source tools is their extensive community support. However, community support is not always helpful when it comes to implementing complex business use cases. A vendor-managed MLOps platform offering expert consultation may be more suitable in this case. But, depending on the quality of their customer support, it can take some time to reach them.

A Word on Open-Source CVOps Tools–Some Common Issues

An open-source CVOps pipeline is like a jigsaw puzzle. If a business can put the correct pieces together, it can achieve a seamless ML development, deployment, and monitoring experience. For many businesses, particularly small to medium-sized, going for open-source tools seems viable due to negligible costs. However, enterprise-scale companies can find it challenging to piece together an end-to-end framework that can scale as per business requirements without causing any significant issues.

Regardless of the company's size and business requirements, open-source MLOps and CVOps present the following problems:

- In general, MLOps open-source tools, and particularly for CVOps, don't offer an end-to-end solution. Some tools may offer data-processing automation, while others may only be used for automated deployments.

- It can become extremely difficult to make multiple open-source tools work together. Therefore, compatibility and interoperability can be some real challenges. Different tools use different types and versions of data and modeling libraries which can be non-compatible with each other. Moreover, open-source tools at times use obsolete packages, which, when discovered, cause more harm than good.

- On the flexibility front, an open-source tool designed to handle, let's say, tabular data cannot be easily modified to handle image data. Even a CV-specialized open-source tool may have missing features like parsing videos and converting them into images. The company employing such tools would have to spend significant resources to implement its desired functionalities.

- Not every data scientist and ML engineer is well-versed in the implementation code of different open-source tools.

Which CVOps Tool Should You Choose? – Custom-Built, Open-Source, or Vendor-Designed

The company's CV architect or executives must make a careful decision based on their business requirements and the resources at their disposal. Though open-source tools are cost-efficient, they present many issues that we have discussed above. Particularly on an enterprise scale, open-source tools are not a viable option.

An enterprise company with enough resources and flexible product delivery can choose to build its custom CVOps tool. However, they must ensure that the tool is flexible, compatible, interoperable, scalable, and user-friendly. They need to document all of their implementation processes to enable ease of use across teams. Often, companies could employ a support team to maintain the tool.

While a custom-built tool is an option, managed CVOps platforms can have more favorable benefits than custom and

open-source platforms. Vendor-managed solutions are fully end-to-end with an integrated pipeline that offers all ML operations from data ingestion to model deployment and monitoring. Also, vendors have specialized personnel working on the operational side, which leads to minimal infrastructure setup, installation, and support issues. An organization simply needs to communicate its requirements to the vendor, and everything else will be taken care of. Moreover, when opting for a managed solution, it is not just the platform that is being bought. Rather, it is a strategic partnership with a vendor that is important for the success of the CV pipeline.

However, with managed solutions, the downside is that the source code is almost inaccessible, and making extensions can be difficult. If the company wants to add an extra feature, then the usual pathway would be to request the vendor. This can take time, and if the fundamental requirements of the ML system change altogether, it would mean that an organization might get locked in, and switching may not be an option.

To avoid such issues, it should be ensured that vendors provide extendible solutions with loosely coupled components. This allows the organization to merge the platform with its own stack to ensure seamless machine learning operations.

Why Picsellia?–A Completely Automated CVOps Stack

Picsellia offers a state-of-the-art, fully managed CVOps platform that includes the following CV operations:

- **Dataset Management**
  Centralize unstructured images and videos to enable greater collaboration and minimize inconsistencies.

- **Experiment Tracking**
  A complete experiment tracking toolkit to facilitate hyperparameter tuning and obtain the best AI model.

- **Model Deployment**
  Immediately put models into a serverless production environment with 24/7 uptime.

- **Model Monitoring**
  Real-time model monitoring to identify edge cases and early detection of model degradation.

- **Automated Pipelines**
  Automate and orchestrate the entire computer vision project lifecycle.

With Picsellia's cost-efficient solution, small, medium, and enterprise companies can significantly reduce ML pipeline failure rates and scale their application as per business requirements.

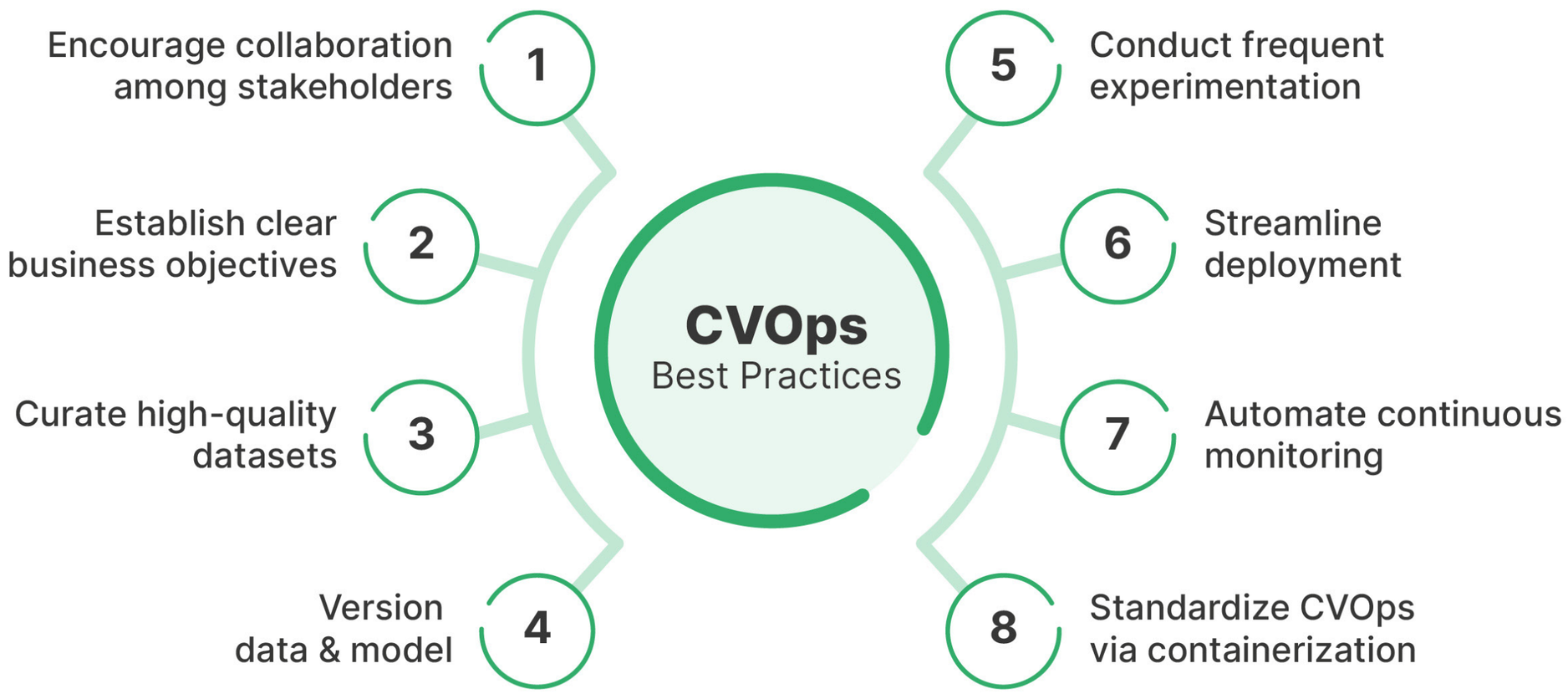
CHAPTER 4

MLOps Best Practices for Computer Vision

A 2020 AI study by MIT Sloan reports that only around 10% of companies obtain substantial monetary benefits from AI and ML technologies. Successful implementation of MLOps can greatly improve the chances of building a financially viable AI application.

MLOps is still under development. With rapid advancements, it can reach maturity within the next few years, offering specialized processes and robust toolchains for computer vision (like CVOps), deep learning (DLOps), and natural language processing domains.

For specialized domains, MLOps pipelines and platforms must be built on consistent principles and best practices to mitigate common challenges and encourage widespread adoption.



An overview of CVOps implementation best practices

Some of the MLOps best practices for computer vision include:

- **Encourage greater collaboration among stakeholders:** In a CVOps pipeline, specialized CV practitioners (engineers, architects, annotators, researchers) are essential in ensuring the project's success. Hence, a collaborative culture that promotes communication between CV-specialized resources and the executives is paramount.

- **Establish clear business objectives:** A CVOps pipeline must have well-defined delivery metrics and KPIs, such as deployment frequency, lead time for changes, mean time to restore, and change failure rate. For instance, the development frequency of a CV application might be slower due to high processing requirements compared to non-CV applications. Hence, application architects must set realistic objectives.

- **Curate high-quality datasets:** A CV pipeline must maintain consistent quality images using image processing techniques like augmentation, cropping, scaling, etc. The dataset must be validated thoroughly before training or testing CV models.

- **Version data and model:** Versioning facilitates AI governance in a CVOps pipeline. It helps auditors trace back different CV models and corresponding image dataset versions to compare results.

- **Conduct frequent experimentation:** CV models are refined using systematic experimentation. A CVOps pipeline with an experiment tracking feature allows practitioners to choose the best model and reproduce results when needed.

- **Streamline deployment:** Using CI/CD techniques, a CVOps pipeline ensures quick delivery of the application. Additionally, robust testing scenarios should be built to ensure that the model behaves as expected at every stage of the deployment process.

- **Automate continuous monitoring:** A CVOps pipeline must continuously monitor the deployed models using performance metrics. If metrics values drop below the desired threshold, the automated pipeline must notify the CV team to mitigate the situation.

- **Standardize CVOps via containerization:** A modularized development process supported by containerization tools (like Docker and Kubernetes) allows more development and deployment flexibility for enterprise CV applications. Containerization makes it easier to automate CVOps workflows. In case of failure, models running in isolated environments would have minimal impact on the remaining application.

While these best practices can be applied across all machine learning domains, they mainly differ in implementation due to domain specific-tasks. The computer vision domain deals with image and video frames; only an exclusive CVOps tool can ensure the reliable application of CV in real-world enterprise applications.

Automate your computer vision workflows with Picsellia!